Docket No. 200210214-1

U.S. Non-Provisional Patent Application

Attorney Docket No.: 200210214-1

Title:

STORAGE ACCESS SYSTEM AND METHOD FOR IMAGE FORMING DEVICE

Inventor:

Brian D. Gragg
12111 Ragweed St.
San Diego, CA 92129
Citizenship: United States of America

STORAGE ACCESS SYSTEM AND METHOD FOR IMAGE FORMING DEVICE

BACKGROUND

[0001] Some image forming devices include storage devices such as memory cards. The memory cards can be accessed through both a universal serial bus mass storage (USBMS) protocol and through a local file system (as when accessed through a CIFS server (Common Internet File System) over a network. USBMS accesses the memory cards at a sector level and puts all file system information on the client host controller such that firmware in the image forming device cannot determine what the USB host is doing to the card. Thus, network access to the card may be potentially corrupted each time the USBMS host writes to the card. In some prior devices, when the memory card was written to by USBMS, the local file system was reset, and any file system clients could be disconnected. With network access to the image forming device, access contentions with the USB host could potentially hang the network computers accessing the memory.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] It will be appreciated that the illustrated boundaries of elements (e.g. boxes, groups of boxes, or other shapes) in the figures represent one example of the boundaries. One of ordinary skill in the art will appreciate that one element may be designed as multiple elements or that multiple elements may be designed as one element. An element shown as an internal component of another element may be implemented as an external component and vise versa.

[0003] Figure 1 is one embodiment of an image forming device configured to control access to a storage device.

[0004] Figure 2 is another embodiment of an image forming device in communication with multiple clients.

[0005] Figure 3 is one embodiment of a storage access manager.

[0006] Figure 4 illustrates one embodiment of a contention matrix defining contention rules.

[0007] Figure 5 illustrates one embodiment of methodology for processing an access request to a storage device.

DETAILED DESCRIPTION OF ILLUSTRATED EMBODIMENTS

[0008] The following includes definitions of selected terms used throughout the disclosure. The definitions include examples of various embodiments and/or forms of components that fall within the scope of a term and that may be used for implementation. Of course, the examples are not intended to be limiting and other embodiments may be implemented. Both singular and plural forms of all terms fall within each meaning:

[0009] "Computer-readable medium", as used herein, refers to any medium that participates in directly or indirectly providing signals, instructions and/or data to one or more processors for execution. Such a medium may take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media may include, for example, optical or magnetic disks. Volatile media may include dynamic memory. Transmission media may include coaxial cables, copper wire, and fiber optic cables. Transmission media can also take the form of electromagnetic radiation, such as those generated during radio-wave and infra-red data communications, or take the form of one or more groups of signals. Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic medium, a CD-ROM, any other optical medium, punch cards, papertape, any other physical medium with patterns of holes, a RAM, a PROM, an EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave/pulse, or any other medium from which a computer, a processor or other electronic device can read. Signals used to propagate instructions or other software over a network, such as the Internet, are also considered a "computer-readable medium."

[0010] "Logic", as used herein, includes but is not limited to hardware, firmware, software and/or combinations of each to perform a function(s) or an action(s), and/or

to cause a function or action from another component. For example, based on a desired application or needs, logic may include a software controlled microprocessor, discrete logic such as an application specific integrated circuit (ASIC), a programmable/programmed logic device, memory device containing instructions, or the like. Logic may also be fully embodied as software.

[0011] "Software", as used herein, includes but is not limited to one or more computer readable and/or executable instructions that cause a computer, processor, or other electronic device to perform functions, actions, and/or behave in a desired manner. The instructions may be embodied in various forms such as objects, routines, algorithms, modules or programs including separate applications or code from dynamically linked libraries. Software may also be implemented in various forms such as a stand-alone program, a function call, a servlet, an applet, instructions stored in a memory, part of an operating system, or other type of executable instructions. It will be appreciated by one of ordinary skill in the art that the form of software may be dependent on, for example, requirements of a desired application, the environment it runs on, and/or the desires of a designer/programmer or the like.

[0012] "User", as used herein, includes but is not limited to one or more persons, software, computers or other devices, or combinations of these.

[0013] In one embodiment, an image forming device is configured with a system to allow an attached storage device, such as a memory card, to be accessed through multiple communication protocols while maintaining the integrity of the storage device. For example, the system can be configured to allow both a universal serial bus (USB) mass storage protocol and a network protocol such as the common internet file system (CIFS) protocol to be simultaneously active and to share the storage device. The system can be configured to control multiple access requests to the storage device in order to reduce or eliminate data corruption. In one embodiment, a control agent (e.g. storage access manager) is configured to control access to the storage device in such a way that corruption is avoided or at least reduced. For example, all clients of the storage device card are made to register with the control agent and request access to the storage device.

[0014] Illustrated in **Figure 1** is one embodiment of an image forming device **100** that is configured to provide access to a storage device **105** and be accessible by multiple communication protocols. The image forming device **100** can be a printer, a copier, an all-in-one product, a multifunctional peripheral, or other device that can form an image onto print media. The image forming device **100** can include various types of imaging mechanisms based on, for example, technologies such as drop-on-demand, ink jet, piezoelectric, thermal printing, laser printing, digital imaging, impact printing, or other available technologies. The storage device **105** can include one or more storage devices and can be embodied as any type of computer-readable medium configured to store data. In one embodiment, the storage device **105** can be a memory card used to store data such as digital images or other types of files.

[0015] The image forming device **100** can include one or more communication ports that allow the image forming device **100** to be in data communication with one or more clients. For example, client A **110** can be in data communication with the image forming device **100** using a communication protocol A **115**. Client B **120** can be connected using a communication protocol B **125** that can be different than protocol A. The protocols may also be the same or a common protocol. In one embodiment, the image forming device **100** can be configured with one or more universal serial bus (USB) ports, one or more network ports, and combinations of these. One or more local clients **130** may also have accessibility to the storage device **105** such as internal functions that can view or manipulate contents within the storage device **105**.

[0016] Since multiple clients can access the storage device **105**, the integrity of the data on the storage device **105** may be affected between simultaneous accesses to the storage device **105**. For example, data can be corrupted when communication protocols do not coordinate with each other, thus, one client can be accessing the storage device unknown to other clients. As used herein, uncoordinating communication protocols include protocols from different devices that may compete for access to the storage device **105** where at least one protocol does not provide notice of the access to the other protocol or device. To control access between such uncoordinating communication protocols, a storage access manager **135** is provided. The storage access manager **135** can be configured to coordinate access requests

between multiple clients. It will be appreciated that uncoordinating communication protocols may be different protocols and/or may be the same protocol. One example of uncoordinating protocols that are the same is where two clients have access to a storage device and both clients use USB protocol. From another perspective, the storage access manager **135** can be regarded as being configured to control access requests from uncoordinating clients or devices.

[0017] In one embodiment, the storage access manager **135** is configured to maintain a current status for the storage device **105** and to determine whether concurrently or simultaneously received access requests are permissible. When the storage device **105** is being accessed by one client, the permissibility of another client simultaneously accessing the storage device **105** can depend on the type of access being requested, the file being accessed, the communication protocol associated with the access request, and/or other desired factors.

[0018] A set of access rules can be predetermined to define permissibility of simultaneous access requests from the multiple communication protocols and types of requests. In one embodiment, the access rules can be configured as a contention matrix or map that indicates what types of simultaneous access requests are permissible in view of a current state of the storage device **105**. In another embodiment, the access rules, also referred to as contention rules, can be programmatically determined by one or more logic algorithms coded with desired conditions that enforce the contention rules. In this manner, the storage access manager **135** receives all access requests from clients and the corresponding client cannot proceed with the access until the storage access manager **135** grants the access.

[0019] If an access request is denied due to a contention, a notification can be sent back to the requesting client denying access. Of course, a variety of options can be configured to handle a denied access request. For example, the request can be denied and the client may be notified to try again at a later time, the access request can be put in a queue and processed when the contention no longer exists, or other type of processing option. With the storage access manager **135**, clients that compete for the storage access device **105** can be coordinated such that all requests are known and monitored. In this manner, each access request goes through the storage access

manager **135** such that each request is approved by the image forming device rather than allowing each client to have direct access.

[0020] Illustrated in **Figure 2** is another embodiment of an image forming device **200** configured to control access to a storage device **205** from multiple clients operating with different communication protocols. In the illustrated embodiment, one or more clients **210** can be connected to the image forming device **200** through a universal serial bus (USB) port using a USB mass storage protocol **215**. The image forming device **200** may also be connected to one or more network clients A-N **220** that communicate through one or more common internet file system (CIFS) servers **225**. Internal clients or functions can also be configured within the image forming device **200** that can access the storage device **205**. The internal clients may include a photo function **230**, a browse drive function **235**, or others. A storage access manager **240** is configured to receive access requests from the various clients and to determine which requests are permissible in order to control access contentions and possible data corruption.

[0021] In the illustrated embodiment, the image forming device **200** is configured such that access requests from the multiple clients are transmitted to the storage access manager **240** for permission. Once access is granted to a client, the client communicates with the storage device **205**. In another embodiment, the storage access manager **240** can be configured to grant access rights to the clients and also control the communication between the clients to the storage device **205**. In one embodiment, the storage access manager **240** is configured as logic such as embedded firmware, downloadable software, or other desired form of logic.

[0022] As previously mentioned, the USB protocol **215** is a sector-level communication protocol that can access data on the storage device **205** by sectors. The CIFS protocol server **225** is a file-level protocol that accesses data on the storage device **205** at a file level. As such, the CIFS protocol server **225** communicates through, for example, an internal file system which is similar to a computer operating system. The internal file system **245** performs the access to the storage device **205**.

[0023] The storage access manager **240** can be configured with one or more contention rules that determine when simultaneous access to the storage device **205** is permissible. One example of the access rules is shown in **Figure 4** as a contention matrix **400** which will be described in greater detail below.

[0024] Illustrated in **Figure 3** is one embodiment of a configuration for a storage access manager **300** configured to control access to a storage device such as a memory card within an image forming device. A communication logic **305** is configured to communicate with a plurality of clients that may compete for access to the storage device with uncoordinating communication protocols. Examples of uncoordinating protocols may include a sector-level protocol vs. a file-level protocol, a sector-level protocol vs. a sector-level protocol, or other types of protocols which include at least one protocol that does not coordinate or otherwise give notice of access requests to other devices or protocols. In general, uncoordinating communication protocols may include protocols where one client can potentially access the storage device without other clients being aware of such access. The storage access manager **300** is configured to provide coordination of access requests from competing clients and to determine which simultaneous access requests are permissible.

[0025] In one embodiment, the storage access manager **300** can include an ID assignment logic **310** configured to assign an identifier to each client/access request received. The identifier can be generated in any desired fashion such as using a random character generator, sequential number assignment, a date-time stamp, combinations of these, or other types of identifiers. In general, the identifier can be configured in any desired manner that allows the storage access manager **300** to process multiple pending and/or active access requests. The identifiers can also be used to associate each access request with its corresponding client such that appropriate messages can be transmitted to the client.

[0026] A contention logic **315** can be included that is configured to determine whether a contention between simultaneous access requests exists. If an access request does not create a contention or otherwise possibly conflict with an active access, the access request would be granted. At that point, the corresponding client

can proceed with accessing the storage device. The contention logic **315** can be configured with access rights or contention rules that define what types of access are simultaneously permissible. In one embodiment, the access rights can be embodied as a contention matrix **320** that defines a plurality of access states for the storage device and whether each state is permissible with each other, or with other defined states. One example of a contention matrix is shown in **Figure 4** as matrix **400**.

[0027] With reference to **Figure 4**, the contention matrix **400** can include one or more possible access states **405** for a storage device that are shown in numeric rows 1-10. The access states **405** can also be considered as possible access operations. Using the configuration shown in **Figure 2** as an example, some possible access states from a variety of communication protocols are shown for the storage device **205**. The illustrated states are based on a configuration having a USB client, a network client such as a CIFS protocol, a photo function that allows the viewing of a proof sheet, and a browse drive function. The storage device in this case is a memory card. Of course, other types and combinations of protocols, access operations, and states can be defined.

[0028] As shown in **Figure 4**, some possible access states **405** include read and write over the universal serial bus (USB) (e.g. rows 1-2), read and write over the network (e.g. rows 3-4), photo access (e.g. row 5), and browse drive access (e.g. row 6). The possible access states are duplicated in the alphabetic columns of the matrix **400** as possible access requests/events **410**. The intersection between the current access states **405** and the access requests **410** form contention matrix values **415** that indicate the permissibility of a simultaneous access. In other words, the contention values **415** define how an access request will be processed based on the current access state for the storage device. Of course, it will be appreciated that other types of access requests and states can be defined based on the types of clients and communication protocols used to communicate with an image forming device. The contention matrix **400** may also include states where multiple USB events may exist from multiple competing USB clients that may compete for access to the storage device.

[0029] With further reference to **Figure 4**, one example of determining the permissibility of an event may be as follows. Suppose the current state **405** of the

storage device is state "2" which is "write over USB." This means that the storage device (e.g. memory card) is currently being accessed by a write command from a first client communicating using the USB protocol. Then suppose another access request is received from another client that is event C "read over network." This means that another client has requested to read data from the storage device and the request was received from a network protocol (e.g. CIFS). The contention logic **315** would determine the current state of the storage device which would be state "2" and determine the permissibility or contention of the access request "C." In this case, the contention value shows a permissibility of "N" which means "no, the event cannot occur." The access request may then be denied and the client requesting such access can be notified. Of course, other options for processing the denied request can be configured such as postponing the request and granting access at a later time.

[0030] As another example, suppose the current access state **405** of the storage device is state "3" which is "read over network." This means that a first client is reading from the storage device over a network connection and protocol. Then suppose an access request is received by the image forming device to write data to the storage device and the request is received from a client connected via the network and using a network communication protocol. The storage access manager **300** would receive the access request, determine the type of request, and determine its contention state with the current state of the storage device. In this case, the type of access corresponds to event "D" which is "write over network." The contention value **415** between state "3" and event "D" is "Y^1" which has been defined as "yes, the event can occur" but it will be allowed only if the access request is for a different file. Thus, if the event D is requesting to write to a different file than the file currently being read from the storage device, the access request will be granted. As mentioned previously, it will be appreciated that other types of contention values/states **415** can be defined for the current access states **405** and the access events **410**.

[0031] With further reference to **Figure 4**, a few contention states **415** include the values "NP" which designate a status of "not possible." In the illustrated embodiment, the "NP" states were defined based on only one USB client being connected to the image forming device. In the illustrated embodiment, it is assumed that the one USB client would not submit multiple access requests to the storage

device simultaneously. Thus, if the current state of the storage device is state "1" where it is being read from the USB protocol, the USB client would not also be submitting read and/or write requests simultaneously. Hence, these contention states **415** are defined as not possible. Of course, the contention states **415** can be defined with multiple USB clients such that the storage device might receive multiple simultaneous access requests from multiple USB protocol devices. In another embodiment, one USB client might be configured to submit multiple access requests so that appropriate contention values **415** would be defined to address those events.

[0032] With reference again to **Figure 3**, the storage access manager **300** can be configured to maintain a current state of the storage device **325** based on currently active access requests that have been granted. When a new access request is received, the contention logic **315** can be configured to determine the current state of the storage device and then determine whether a contention would occur from the received request based on the contention rules **320**. Once the permissibility of the access request is determined, a notification logic **330** can be configured to transmit a message to the requesting client. The message can notify the requesting client, for example, that access is granted, denied, postponed, or other predefined notice.

[0033] It will be appreciated that the contention matrix **320** can be configured in a variety of ways. For example, the defined contention states and/or conditions may be programmatically determined based on a contention algorithm that can determine the permissibility of simultaneous access requests based on predetermined allowable access conditions. In another embodiment, the contention matrix **320** can be embodied in different forms such as a map, a table, or other data structure that allows for the determination of the access conditions and contention states.

[0034] Illustrated in **Figure 5** is one embodiment of a methodology **500** for processing multiple access requests in an image forming device. The illustrated elements denote "processing blocks" and represent software instructions or groups of instructions that cause a computer or processor to perform an action(s) and/or to make decisions. Alternatively, the processing blocks may represent functions and/or actions performed by functionally equivalent circuits such as a digital signal processor circuit, an application specific integrated circuit (ASIC), or other logic device. The diagram,

as well as the other illustrated diagrams, do not depict syntax of any particular programming language. Rather, the diagram illustrates functional information one skilled in the art could use to fabricate circuits, generate software, or use a combination of hardware and software to perform the illustrated processing. It will be appreciated that electronic and software applications may involve dynamic and flexible processes such that the illustrated blocks can be performed in other sequences different than the one shown and/or blocks may be combined or, separated into multiple components. They may also be implemented using various programming approaches such as machine language, procedural, object oriented and/or artificial intelligence techniques. The foregoing applies to all methodologies described herein.

[0035] With reference to **Figure 5**, when a request to access a storage device on an image forming device is received, the process is initiated to determine whether the access is permissible (**Block 505**). In one embodiment, each client and/or access request is assigned an identifier to distinguish it from other clients/requests (**Block 510**). In another embodiment, assigning an identifier may not be required depending on, for example, the communication architecture, protocols, or other configurations. Based on the type of received access request, a contention status of the storage device is determined (**Block 515**) and a determination is made whether the access is permissible (**Block 520**).

[0036] As discussed in a previous embodiment, a set of access rights or contention rules can be predetermined to define the permissibility of simultaneous access to the storage device. The simultaneous access can be defined based on, for example, the type of access requested and the type of communication protocol associated with the access request. Of course other types of factors can be used to define the contention rules. In one embodiment, the contention rules can be defined according to a contention matrix similar to the matrix **400** shown in **Figure 4**. A set of possible access states can be defined based on types of access requests and types of communication protocols used. For each possible type of access defined, a contention status/value can then be defined with all the types of access.

[0037] When an access request is received, it is compared to the current access state of the storage device to determine whether a contention will be created if the

received access request is granted. If access is not permissible (**Block 520**), the requesting client is notified and access is not granted (**Block 525**). In one embodiment, the denied request can be paused and allowed at a later time once the contention no longer exists. The denied request may also be simply terminated where the client would have to re-submit the request at a later time. Of course, the manner in which an access request is processed can be defined within the contention rules that define different states such as an unconditional access, conditional access, no access, or other types of conditions. Examples of some possible contention conditions are shown in **Figure 4**.

[0038] If access is permissible at **Block 520** (e.g. the access request will not create a contention with the current state of the storage device), the contention status/current state of the storage device is updated to reflect the simultaneous access (**Block 530**). Access can then be granted to the requesting client (**Block 535**). The process can then return until another access request is received or can terminate until reinitiated by a new access request.

[0039] With the described embodiments, an image forming device can be configured to allow a storage device(s), such as one or more memory cards, to be accessed through multiple protocols, one or more of which may be an uncoordinating protocol, while maintaining card integrity. For example, both the USB Mass Storage protocol and the network CIFS protocol can be simultaneously active and able to share the memory card device.

[0040] Suitable software for implementing the various components of the present system and method using the teachings presented here include programming languages and tools such as Java, Pascal, C#, C++, C, CGI, Perl, SQL, APIs, SDKs, assembly, firmware, microcode, and/or other languages and tools. The components embodied as software include computer readable/executable instructions that cause one or more computers, processors and/or other electronic device to behave in a prescribed manner. Any software, whether an entire system or a component of a system, may be embodied as an article of manufacture and maintained as part of a computer-readable medium as defined previously. Another form of the software may include signals that transmit program code of the software to a recipient over a

network or other communication medium. It will be appreciated that components described herein may be implemented as separate components or may be combined together.

[0041] While the present invention has been illustrated by the description of embodiments thereof, and while the embodiments have been described in considerable detail, it is not the intention of the applicants to restrict or in any way limit the scope of the appended claims to such detail. Additional advantages and modifications will readily appear to those skilled in the art. Therefore, the invention, in its broader aspects, is not limited to the specific details, the representative apparatus, and illustrative examples shown and described. Accordingly, departures may be made from such details without departing from the spirit or scope of the applicant's general inventive concept.